Tampere University

Jesper Granat

LOCATING KNOWN OBJECTS USING TWO CAMERAS

Faculty of Information Technology and Communication Sciences
Bachelor of Science Thesis
May 2019

ABSTRACT

Jesper Granat: Locating Known Objects Using Two Cameras
Bachelor of Science Thesis
Tampere University
Electrical Engineering
May 2019

Locating objects is a common problem in geometric computer vision. The goal is to determine the coordinates of an object in a scene. When the object is projected into the camera plane, the depth information is lost, and to overcome this, two cameras are used. Several approaches can be found in the literature on how to reconstruct a scene with two cameras. Binocular stereo vision tries to emulate the human vision by placing the two cameras next to each other, and other methods, such as the one studied here, use cameras far away from each other and in a steep angle to each other.

This thesis investigates the latter cae. Due to the large difference in what the cameras can see, a full reconstruction of the scene would be difficult. The method used in this study uses a fully calibrated approach where two cameras are calibrated both individually and relative to each other. With the known relative positions of the cameras in addition to the internal parameters of both cameras, points of interest on both images can be triangulated into the 3D space. The points are selected based on the corners of the bounding boxes of the objects, which are detected using state-of-the-art Mask R-CNN object class and segment detector. With this method, a bounding box is determined in 3D.

Finally, the empirical results of the method and the effects of long separation of the cameras are presented. According to the evaluation, the 3D localization accuracy improves with camera separation. The evaluation shows that the calibration method also affects the accuracy of the localization and with errors in calibration the final bounding box will not be consistent.

Keywords: Computer vision, Object detection, Camera calibration, Triangulation, Epipolar geometry

PREFACE

The topic of this thesis was given by my employer, namely *Wapice Ltd*. The goal was to investigate methods and literature on 3D localization and reconstruction that could be utilized in a wide array of applications of geometric computer vision. Because of the complexity of the task, the topic was restricted to its current form.

I would like to thank Ilari Kampman from Wapice for providing the topic and for supporting me in the research and implementation processes. I am thankful for my colleagues at Wapice for helping me in practical matters and for giving me feedback to improve on. I am equally grateful for my supervisor Esa Rahtu for helping me refine the topic and providing me generous guidance throughout the thesis.

Tampere, 6th May 2019

Jesper Granat

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROGRAMS AND ALGORITHMS

LIST OF SYMBOLS AND ABBREVIATIONS

BRIEF	Binary Robust Independent Elementary Features, a keypoint descriptor
CNN	Convolutional Neural Network
FAST	Features from Accelerated Segment Test, a keypoint detector
Mask R-CNN	Masking Regional Convolutional Neural Network
ORB	Oriented FAST and Rotated BRIEF, a keypoint detector and descriptor
RANSAC	Random Sample Consensus, an algorithm for removing outliers
RMS	Root Mean Square, square root of the sum of the squared means
RoI	Region of Interest, a spatial region in an image where an object might be located
RPN	Region Proposal Network, a neural network for RoI extraction
K	Calibration matrix containing the intrinsic parameters
E	Essential matrix
f	Focal length
F	Fundamental matrix
P	Projection matrix
R	Rotation matrix between two coordinate systems
θ	Angle of rotation
$\mathbf{t}$	Translation vector between two coordinate systems

1 INTRODUCTION

Using two cameras to locate an object is a fundamental problem in geometric computer vision. Geometric computer vision studies the shapes and size of the scene with respect to the cameras. [10, p. xi] There are several applications for object localization both in industry and in entertainment. One of the modern applications is in self-driving cars where understanding the distances in the scene is crucial [17][22]. In the industry, 3D localization can be applied to, for example, detecting workers in construction zones to aid site management in organizing work [15]. Because of the introduction of neural networks for image processing, the possibilities of object localization have been extended.

In this study neural networks are utilized to find bounding boxes for a known object, and using the knowledge of the relative location of the two cameras, the object can be located. Traditionally in binocular stereo vision, the cameras have been placed next to each other, or otherwise computationally rectified. Additionally, stereo correspondences are used for a full 3D reconstruction. The goal of this paper, however, is to design and implement a robust method for large scale stereo vision, and therefore the cameras are placed far from each other and pointing in widely different directions. Hence, it would be difficult to utilize a full 3D reconstruction. An alternative approach is studied where only a small set of points is reconstructed, and these points then define a bounding box in 3D for the object.

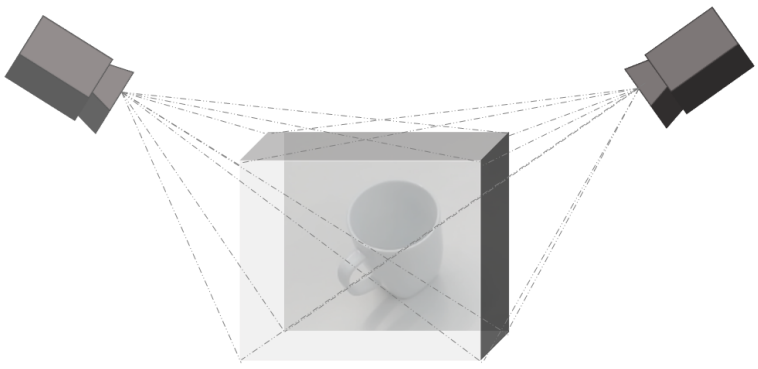


Figure 1.1. *The objective of the study. Corners of bounding boxes from both cameras are triangulated into the 3D space to form a bounding box. There is assumed to be only one object, and its class is assumed to be from a pre-defined set.*

This study studies a method that can be used to locate an object that is seen by both cameras. The studied method is an application of the pipeline presented by Hartley and

Zisserman [10, p. 262]. Firstly, the cameras are calibrated. The intrinsic parameters, the focal length and the principal point, of both cameras are calibrated using a set of images of a calibration pattern. This knowledge is then used to calibrate the camera extrinsic parameters, which describe the camera position in the 3D world. Once the cameras are calibrated, objects from a pre-defined set of classes are detected from the scene, and their corners are triangulated into the 3D space to determine the bounding volume of the object. This is illustrated in Figure 1.1. The studied method can be extended to several objects, but that is out of the scope of this study.

This document is structured as follows. Chapter 2 discusses briefly the works of others in the field of stereo vision and introduces the background knowledge on the area. The stereo vision pipeline is also introduced theoretically. Chapter 3 describes how the camera setup is constructed and how the pipeline could be implemented. Chapter 4 discusses the obtained results of this study. Finally, Chapter 5 summarizes the study and presents the conclusions.

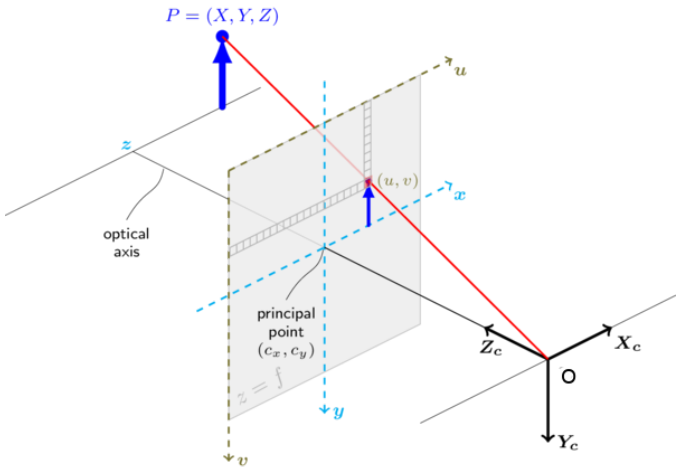
2 BACKGROUND

The pipeline studied in this paper is constructed by detecting a bounding box for an object from both camera views, and then triangulating the corners of the bounding box into the 3D space. In order to detect the bounding box, an object detector is needed. Additionally, in order to triangulate the corners, knowledge of the relationships between the cameras and the 3D world is needed. This process is called the camera calibration. In camera calibration, the relative positions and orientations of the cameras are measured. Also the relationship between pixel coordinates and 3D coordinates is measured in camera calibration. Additionally, camera calibration utilizes feature extraction in order to detect either a calibration object or keypoints from the calibration image.

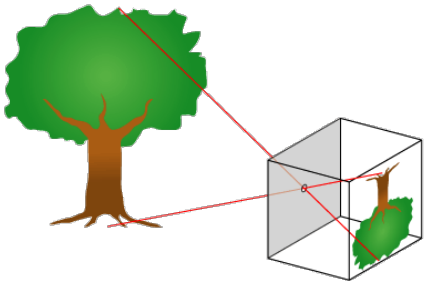
To understand the whole pipeline, several models and methods are needed. They are then combined to form the pipeline. First, to understand how a single camera is calibrated, the pinhole camera model is introduced. Then, to understand how two cameras can be related, epipolar geometry is briefly explained. From these concepts triangulation is introduced. Then feature extraction and matching is described which both can be used in camera calibration. After these, the camera calibration is explained. Finally, object detection is covered.

2.1 Camera Model

The camera model used in this study is called the pinhole camera model. It is used here, and augmented with the radial distortion model. In an ideal pinhole camera, a light ray from a point in an object is directed through an infinitely small hole and is captured on the image plane as a single point. This happens for all points of an object that are in a direct line of sight of the camera. In this ideal model the image will be formed upside down on the image plane, as shown in Figure 2.1b. To turn this concept into a mathematical model, the image plane is moved in front of the pinhole. In this way the image will be right way up. This mathematical model is illustrated in Figure 2.1a. Distortion, on the other hand comes from the camera lens, and therefore its cause or effect is not visible in the figures. In distortion, lines that are straight in the 3D world are not straight in the image, and specifically in radial distortion those lines will be curves in the image. [10]



(a) *The geometry of the pinhole camera model [26], notation adapted.*



(b) *The idea behind the pinhole camera model [21].*

Figure 2.1. *The pinhole camera model.*

The mapping $\mathbf{X} \to \mathbf{x}$ from world coordinates to image coordinates can be written as

$$\underbrace{\begin{bmatrix} X & Y & Z \end{bmatrix}^T}_{\mathbf{X}} \to \underbrace{\begin{bmatrix} fX/Z & fY/Z \end{bmatrix}^T}_{\mathbf{x}} = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.1}$$

where f is the focal length of the camera, X, Y and Z are world coordinates, and x and y are image coordinates. It becomes more useful to represent these coordinates in homogeneous coordinates. In homogeneous coordinates an additional dimension is added to the coordinate system. This means that the scale of homogeneous coordinates is arbitrary, and all points that are scalar multiples of another point, are in fact the same point in euclidean coordinates. Intuitively this means that all scalar multiples of a point lie on the same line. For more information, see [10]. In the homogeneous representation this mapping can be expressed as

$$\underbrace{\omega \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\mathbf{X}} \tag{2.2}$$

where P is called the projection matrix and ω is an arbitrary scale that comes from the conversion to the homogeneous coordinates. This equation maps the world coordinates into coordinates on the image plane. Equation 2.2 can be expressed concisely as

$$\mathbf{x} = P\mathbf{X}. \tag{2.3}$$

In order to represent the camera coordinates in terms of pixels, the focal length is multi-

plied by the known pixel densities in both directions, and the center of the image plane is shifted to the image center in terms of pixels. In this study, the origin of the image coordinates is considered to be in the top left-hand corner, with y increasing downwards and x increasing to the right as shown in Figure 2.1. Therefore, in the ideal case the principal point, the point where the optical axis intersects the image plane, would be in the middle of the image. If the optical center is not in the center of the image, that can be accounted by offsetting the principal point. Additionally, if the camera is not located at the origin of the world coordinates, or if the camera coordinates are not aligned with the world coordinates, the translation and rotation of the camera need to be accounted for. This can be done by doing a similarity transformation from the world coordinates to the camera coordinates. This is done by multiplying by the combination of a rotation matrix and a translation vector. By adding these extensions to the camera model, Equation 2.2 becomes

$$\underbrace{\omega \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{[R|\mathbf{t}]} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\mathbf{X}} \tag{2.4}$$

where u and v represent the image coordinates in terms of pixels, r_{ij} represent the components of the rotation matrix, t_i represent the translation vector, f_x and f_y are the focal lengths multiplied by the pixel densities in x- and y-directions respectively, and c_x and c_y are the x- and y-components of the principal point respectively. By denoting

$$P = K[R|\mathbf{t}] \tag{2.5}$$

Equation 2.3 still applies, but this time the projection matrix P accounts for pixel values, principal point as well as rotation and translation. Additionally, $\mathbf{x}$ will be in terms of pixels. The matrix K is known as the camera intrinsic matrix, and the matrix $[R|\mathbf{t}]$ is known as the extrinsic matrix. [10][26]

In addition to the effect of the principal point, focal length and pixel densities, real cameras usually have at least some degree of distortion. Generally distortion is divided into radial distortion, tangential distortion and thin prism distortion. In this study it is assumed that there is no tangential or thin prism distortion because measuring them would require more accurate calibration equipment. Radial distortion means that straight lines in 3D are curved in the image. The lines can be curved towards or away from the center of the image. If the lines are curved away from the center, it is said to be barrel distortion, and in the other case pincushion distortion. [26][10][32]

Correcting radial distortion can be done after the image has been taken. It is not necessary to correct the whole image but only the points considered in each case. In fact, undistorting the whole image and then doing analysis on it may cause more errors than first analyzing it and later correcting the distortion. This is because undistortion introduces

errors in the noise model and may cause aliasing. [10] Radial distortion is corrected after transformation to the image plane but before conversion to pixel coordinates. Radial distortion is corrected with the equations

$$\begin{aligned} \hat{x} &= x\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} \\ \hat{y} &= y\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} \end{aligned} \tag{2.6}$$

where $\hat{x}$ and $\hat{y}$ are the radial distortion corrected coordinates, k_1, k_2, k_3, k_4, k_5 and k_6 are the radial distortion coefficients, and $r^2 = x^2 + y^2$ is the radius of the radial distortion. In a simpler model some of the coefficients could be left out from the largest index to smallest. [26][32] In that case they are set to zero in Equation 2.6.

2.2 Epipolar Geometry

The discussion in this section is based on [10]. The internal parameters of a camera define how the camera is related to the 3D world in terms of the coordinate system of the camera. However, with two cameras it is essential to know how the two cameras are related to each other. In this study, the left camera coordinate system is adopted as the world coordinate system. This means that the world origin is at the left camera center, and the orientation of the left camera coordinate system defines the orientation of the world coordinate system. Now, the location and orientation of the right camera is measured with respect to the left camera. The goal of epipolar geometry is to understand the relationships between two images from two cameras. The relationship can be expressed in terms of rotation and translation between the cameras, and together with the intrinsic parameters, the relationship between the two images is known.

There exists a restriction on where two image points can lie on their respective images, given that they originate from the same 3D point. This restriction intuitively rises from the fact that the cameras are rotated and translated with respect to each other, and therefore two image points from the same 3D point cannot lie anywhere on the images. The relationship between two images can be expressed with the fundamental matrix F, which puts a constraint on the locations of corresponding points between the images. This restriction is mathematically defined as

$$\mathbf{x}_2^T F \mathbf{x}_1 = 0 \tag{2.7}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are homogeneous image points in the right and left image respectively. This constraint holds for corresponding image points. When dealing with camera calibration, it can be useful to have a similar constraint on normalized coordinates. Normalized image points can be thought as image points of an image that was taken with a camera that has the identity matrix as its intrinsic calibration matrix K. Normalized image points can be calculated from image points by multiplying by the inverse of K such as

$$\hat{\mathbf{x}} = K^{-1}\mathbf{x} \tag{2.8}$$

where $\hat{\mathbf{x}}$ is $\mathbf{x}$ converted into normalized image coordinates. A matrix that has a similar property as F is called the essential matrix E. Unlike F, E operates on normalized image coordinates. It is related to F with

$$E = {K_2}^T F K_1 \tag{2.9}$$

where K_1 and K_2 are the camera intrinsic matrices of the right and left camera respectively. Sometimes, especially in camera calibration (see Section 3.2), E is known before F. Then F can be solved with

$$F = {K_2}^{-T} E {K_1}^{-1}. \tag{2.10}$$

The essential matrix has a similar constraint as the fundamental matrix in the form of

$$\hat{\mathbf{x}}_2^T E \hat{\mathbf{x}}_1 = 0 \tag{2.11}$$

where $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are corresponding normalized homogeneous image points. With the constraints on E and F, a point on one image can be mapped into a line on the other image. The exact location of the corresponding point, given E or F and one point, is not known. This is because of the lost depth knowledge when the points in the 3D world were projected into the image. Therefore both matrices set a constraining line on the other image where the corresponding point must lie. This constraining line is called the epipolar line, and the constraint is known as the epipolar constraint. This is visualized in Figure 2.2 where a point on the left image is mapped to a line in the right image.

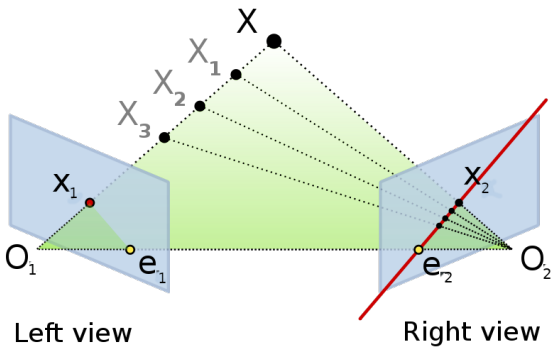


Figure 2.2. *Epipolar geometry between two cameras [25], notation adapted.*

The essential matrix can also be calculated directly by knowing the relative rotation and translation between the cameras. This is done with

$$E = [\mathbf{t}]_x R \tag{2.12}$$

where $[\mathbf{t}]_x$ describes the skew symmetric matrix of the translation $\mathbf{t}$ from left camera coordinates to right camera coordinates. Similarly R is the rotation from left camera coordinates to right camera coordinates. Together R and $\mathbf{t}$ form a similarity transformation

from the left camera view to the right. The skew symmetric matrix is defined as

$$[\mathbf{a}]_x = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.13}$$

where $\mathbf{a}$ is a vector of length 3, and a_1, a_2, a_3 are its components. This is the matrix form of the vector cross product. The essential matrix is homogeneous which has the consequence that the scale of the translation is lost. It is also possible to decompose E into rotation and translation. This is done in Section 3.2.

2.3 Triangulation

By knowing the camera intrinsic and extrinsic parameters, and corresponding points from both images, it is possible to determine the 3D point that was mapped into the corresponding image points. This process is known as triangulation, and it is in a key role when the bounding volume is determined. So far the discussion has been about mapping 3D points into image points but in triangulation the goal is the opposite. The description in this section is based on [10].

The point of intersection is determined by solving a homogeneous system of equations that arise from the constraint between world points and image points in Equation 2.3. The problem is formulated in the form of a linear equation

$$A\mathbf{X} = \mathbf{0} \tag{2.14}$$

where A is a weight matrix and $\mathbf{X}$ is a point in 3D. The weight matrix is constructed from two constraints on the image relationships $\mathbf{x_1} = P_1\mathbf{X_2}$ and $\mathbf{x_2} = P_2\mathbf{X_2}$. Both of these can be represented with a cross-product such as $\mathbf{x} \times (P\mathbf{X}) = 0$. This can be written out as

$$\begin{aligned} x(\mathbf{p}^{3T}\mathbf{X}) - \mathbf{p}^{1T}\mathbf{X} &= 0 \\ y(\mathbf{p}^{3T}\mathbf{X}) - \mathbf{p}^{2T}\mathbf{X} &= 0 \\ x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) &= 0 \end{aligned} \tag{2.15}$$

where x and y are the two components of $\mathbf{x}$, and p^i is the row at index i of P. Of these equations only the two first are linearly independent as the third equation is dependent on x and y already present in the two first equations. Now A in Equation 2.14 can be written as

$$A = \begin{bmatrix} x_1\mathbf{p_1}^{3T} - \mathbf{p_1}^{1T} \\ y_1\mathbf{p_1}^{3T} - \mathbf{p_1}^{2T} \\ x_2\mathbf{p_2}^{3T} - \mathbf{p_2}^{1T} \\ y_2\mathbf{p_2}^{3T} - \mathbf{p_2}^{2T} \end{bmatrix}. \tag{2.16}$$

The algorithm for solving triangulation from this system of equations is presented in Algorithm 2.1. The algorithm is the least squares solution to the homogeneous system of equations.

```
function triangulate(image_points, P1, P2):
    # compose A as presented in equation 2.16
    A = compose_weights(image_points, P1, P2)
    # compute the SVD of A
    U, D, V = SVD(A)
    # solution is in the last column of V
    point = V[-1, :]
    return point
```

Algorithm 2.1. *Triangulation.*

2.4 Feature Detection and Matching

In order to calibrate the cameras, corresponding points from both images are needed to be extracted. The discussion in this section is based on two cases. In the first case there is a known pattern in the image. Such a pattern might be, for example, a chessboard. Another case is where there is no known pattern in the image. In that case, feature and descriptor extraction is needed. This latter case will only be considered shortly.

One of the most commonly used calibration patterns is a chessboard pattern [19], such as the objects in Figure 4.1. Another calibration pattern is a cube whose 3D geometry is well known. In this case one image would be enough. Such equipment, however, is often expensive and requires more set up than a simple 2D pattern. [35] Therefore, a 2D chessboard pattern is used in this study. The calibration with a 2D pattern requires knowing the 2D geometry of the pattern, and the corresponding geometry is needed to be detected from the image. In the case of the chessboard, the geometry is a grid of points. The dimensions of the grid are the same as the dimensions of the inner corners of the chessboard. The challenge is then to detect those inner corners from the image. The method introduced here is from OpenCV library [2], see Chapter 3. In this method, briefly, the image is first binarized via thresholding in order to segment the white and black squares. Secondly, the corners of the black squares are detected. Thirdly, quads are extracted from the image, and finally the quads are grouped into lines based on the 2D geometry of the calibration pattern. [34]

On the other hand, general feature extraction does not require a specific pattern with a known geometry in the image. In this case, points of interest or keypoints are detected from distinguishable places in the image. Such places might include corners or peaks that are uniquely described by their orientation and edge profile [32]. One algorithm for feature and descriptor extraction is ORB (Oriented FAST and rotated BRIEF) [31]. It is based on the FAST (Features from Accelerated Segment Test) keypoint detector [30] and the BRIEF (Binary Robust Independent Elementary Features) keypoint descriptor [3].

The following description is based on [31]. ORB is a fast and robust algorithm that performs well under noise. It adds the orientation component to its features making it sustain rotation between the images which is suitable for this method. First the keypoints are detected with FAST, and then filtered with Harris corner measure [9], and scale is accounted for with a scale pyramid, see for example [32, p. 144]. The orientation is measured with intensity centroid, see [28]. The descriptors are then obtained using rotational BRIEF which is a modified version of BRIEF where the rotation is accounted for by steering BRIEF according to the keypoint orientation. This means that the features are rotated according to their orientation. The features are vector of n binary tests. However, the rotation reduces the variance among the features which makes the matching process more difficult. In addition, the correlation between the tests increases with rotation. The loss of variance and gain of correlation are compensated by a greedy search for binary tests with high variance and low correlation from a training set of all possible features. For more details, see [31]. An example of keypoints detected by ORB is shown in figure 2.3.

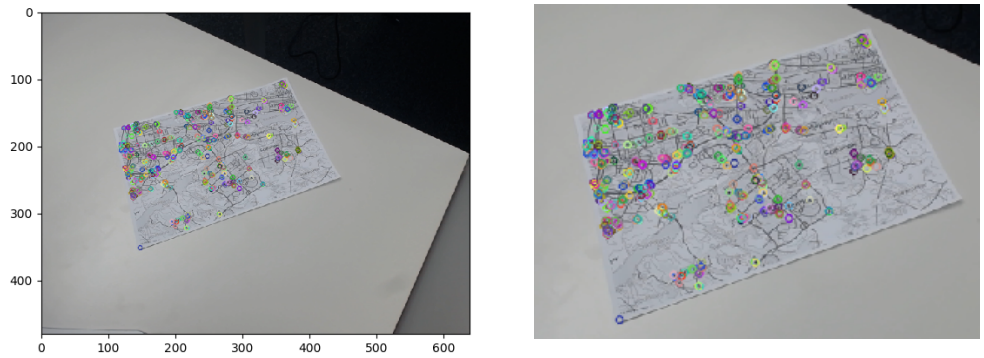


Figure 2.3. *Keypoints detected by ORB. A map contains several unique features, and therefore it is a good scene for keypoint extraction. The left-hand side image shows the detected keypoints drawn on the original input image of the algorithm, and the right-hand side image is a zoomed in version of the left image.*

Now that the keypoints are detected, they need to be matched. In the case of a calibration pattern, the keypoints are by default in the same order. On the other hand, for ORB keypoints, matching is required. The keypoints themselves do not contain enough information to be uniquely matched, and therefore the extracted descriptors are used. As explained, the ORB features have high variance, and therefore a simple nearest neighbor search can be used. In order to match the keypoints it is enough to compute the distance of every descriptor in one camera to every descriptor in the other camera. Then the descriptors that are the closest to each other form a match. This is formulated as

$$\boldsymbol{x}' = \underset{\forall \boldsymbol{x}_n \in D}{\arg\min}\, d(\boldsymbol{x}, \boldsymbol{x}_n) \tag{2.17}$$

where $\mathbf{x}'$ is the match for the feature $\mathbf{x}$, D is the set of features in the other image, and d is a distance function. Euclidean distance is used as a choice for the distance function. For a vector of length n, such as the feature vector recovered from ORB, the distance d

is calculated as

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{M} (a_i - b_i)^2} \tag{2.18}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the feature vectors.

2.5 Calibration

The following discussion is based on [10], [32] and [13]. Camera calibration consists of two different calibrations — the intrinsic camera parameters and the extrinsic camera parameters. Firstly, the intrinsic parameters are described in a calibration matrix K, which consists of focal lengths and the pixel dimensions in both directions. The intrinsic parameters also include the distortion coefficients.

To calibrate the individual camera parameters the method developed by Zhang [35] is used. To summarize, the first step is to show a planar calibration pattern from at least two separate angles. However, more images from more angles reduce the effect of noise on the calibration accuracy. Then feature points are detected from each image, and intrinsic and extrinsic parameters are estimated using a closed form solution [35, section 3.1]. Then the distortion parameters are estimated using the least squares method. Finally, all parameters are optimized using complete maximum likelihood estimation [35, section 3.3]. The overall error could be reduced by a factor of 2–3 by including distortion in the camera calibration. Additionally, the estimated effective focal length might change due to distortion. [10] The advantage of Hang's method is that it gives also the camera pose relative to the calibration object.

From the intrinsic and extrinsic parameters of each individual camera, it is easy to calculate the relative stereo parameters. The extrinsic parameters are presented in the extrinsic matrix $[R|\mathbf{t}]$. In some implementations of the method in [35], the rotation is given as a vector instead of a matrix. The conversion between these two can be done with the equation

$$R = \cos\theta I + (1 + \cos\theta)\mathbf{r}\mathbf{r}^T + \sin\theta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{2.19}$$

where R is the rotation matrix, **r** is the rotation vector with r_x, r_y and r_z its components and $\theta = ||\mathbf{r}||$ is the angle of rotation [26].

Another method of interest involves feature extraction and five-point algorithm introduced by Nister in 2004 [23]. The following discussion is based on the original paper and [16]. The five-point algorithm estimates E from the two sets of corresponding image points and the calibration matrix. Each corresponding pair of points $\mathbf{x_1^i}$ and $\mathbf{x_2^i}$ together with the constraint in Equation 2.11 can be formatted into a 5×9 matrix M where the column at i is

$$M_i = [x_1^{i,1}x_2^{i,1}\ x_1^{i,2}x_2^{i,1}\ z_1^{i,3}x_2^{i,1}\ x_1^{i,1}x_2^{i,2}\ x_1^{i,2}x_2^{i,1}\ x_1^{i,3}x_2^{i,2}\ x_1^{i,1}x_2^{i,3}\ x_1^{i,2}x_2^{i,3}\ x_1^{i,3}\ x_2^{i,3}]^T \tag{2.20}$$

where $x^{i,j}$ are the three components of $\mathbf{x^i}$. This matrix has a null-space representation with four vectors that can be presented with four matrices X, Y, Z and W, and then E has the form

$$E = xX + yY + zZ + wW \tag{2.21}$$

where x, y, z and w are scalars and $w = 1$ because the coordinates are in homogeneous format. Hence, this algorithm can determine E only up to an arbitrary scale. Now, inserting Equation 2.21 into equations 5 and 6 in [23], a system of equations can be formulated. It will have a monomial basis of

$$[x^3 \ y^3 \ x^2y \ xy^2 \ x^2z \ x^2 \ y^2z \ y^2 \ xyz \ xy \ x \ y \ 1] \tag{2.22}$$

where each equation of the system comes from the ten constraints of E. Denoting this system with A and applying Gauss-Jordan elimination on A, it can be reduced into an upper triangle form. Taking the last three columns of A and using equations 11–13 in [23], a new 3×3 matrix B representing a system of equations in terms of z can be formed. The determinant of this is a tenth degree polynomial n. For each of the roots of n (in terms of z), the corresponding x and y can be solved from B. Now, E can be constructed with Equation 2.21, and Section 3.2 describes how R and $\mathbf{t}$ can be recovered from E. In order to survive outliers, the RANSAC (Random Sample Consensus) algorithm [7] should be used. This algorithm modified for five-point algorithm is outlined in Algorithm 2.2. Here five points, the minimum number of points needed to fit the model, are used to generate a hypothesis for E. This is repeated several times and the best hypothesis is selected. The measure for the best hypothesis can be the number of outliers, or pre-emptive scoring [24] can be used.

```
1 function five_point_with_RANSAC(matches, N):
2     points = matches[0:5]
3     best_E = five_point(points)
4     for each i in [0, N]:
5         points = matches[random_integers(5) % matches.size]
6         E = five_point(points)
7         if E is better hypothesis than best_E:
8             best_E = E
9     return best_E
```

Algorithm 2.2. *RANSAC for the five-point algorithm.*

2.6 Object Detection

Object detection is a fundamental problem in computer vision. This study utilizes Mask R-CNN [11]. It detects a bounding box, segments a mask and predicts a class probability for each object in the image.

Mask R-CNN is a convolutional neural network (CNN) which means that it consists of

several layers of convolutions [11]. The convolution weights and biases are learned from training data. Modern convolutional neural networks use several layers. A general architecture consists of blocks of convolutional layers, non-linear activation layers and a max-pooling layer. In the convolutional layer, the input blob is convolved with a filter. Then, each cell is activated based on the output of the activation layer. Finally, to reduce the size of the blob only the maximum value of a layer is selected. The learning of the CNN happens by minimizing the error function. After each iteration, the weights of the convolutional filters are adjusted towards the negative gradient of the error function. [14] The Mask R-CNN architecture is presented in Figure 2.4.

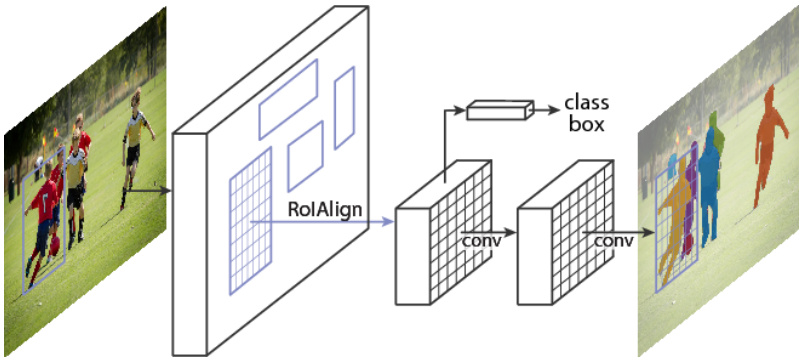


Figure 2.4. *Mask R-CNN architecture [11].*

The following description is based on [11], [27] and [8]. Mask R-CNN is a revised version of Faster R-CNN [27] which outputs the object segmentation in addition to the object class probabilities and bounding boxes. Faster R-CNN consists of two concurrent networks, namely a region proposal network (RPN) that proposes object agnostic regions of interest (RoI) where objects might be located, and a classification network that classifies the possible object and refines the bounding box. The RoI are pooled using a RoIPool layer [8] where each region of interest is quantized into smaller feature maps using max pooling while preserving the spatial layout.

Mask R-CNN extends this idea by running a third network in parallel with the classification network that proposes a segmentation of the object instance. The mask from a RoI is predicted with a fully convolutional network (FCN) [20] which allows the region to maintain its 2D layout instead of having to be vectorized.

3 IMPLEMENTATION

Camera set up includes two parts. The first part is deciding the physical locations and angles of the cameras. The second part is calibrating the intrinsic and extrinsic parameters of each camera. Both of these phases are only required once as long as the cameras do not move or their intrinsic parameters do not change, for example due to changes in the temperature. Therefore the computational complexity of this step is not crucial to the overall performance of the program when in use.

With the system set up, the cameras are ready to capture images of their surroundings. From each image all objects are first detected with Mask R-CNN. For the scope of this study, each frame is assumed to contain only one known object. The version of Mask R-CNN used here is trained with the Microsoft COCO data set [18], and the known objects are assumed to be those in the data set. From the bounding boxes generated by Mask R-CNN the bounding volume is determined via triangulation of the corners of the bounding boxes.

The program is implemented in Python [29] using the scientific Python libraries [12] and OpenCV [2]. These tools are chosen because they are open source and portable. Python is a flexible programming language and is suitable for testing different solutions as its syntax is concise and it does not require a long compilation process. OpenCV, in turn, is a widely used computer vision library that has a large community and it was originally developed for C++. Its Python interface offers bindings to their C++ counterparts, and hence it is easy to convert programs written in Python into C++ for production use.

3.1 Physical Set Up

The main concern of this study is to investigate the challenges presented by the long baseline and the high angle between the principal axis of the cameras. Therefore the cameras are located accordingly. In this way the problems that arise from the increase of the baseline can be investigated. Also the angle is increased, and its effects are studied.

The cameras for this study were mounted high from the ground level to allow them to have as many common points as possible. The most common problem that comes from the difference in physical location is the inclusion of occlusions, which prevent the two cameras from seeing the same points. Therefore it helps to have the cameras relatively high from the ground level. On the other hand, object detection becomes more difficult as the cameras are higher, as objects are usually detected near the eye-level. Of course,

with more train data this will not be a problem. That, however, is beyond the scope of this study. As in all computer vision applications that utilize static cameras, the cameras are safely fastened to their mounts to keep them from changing position or pose even when there are some external disturbances. The lenses are cleaned and the lighting is even.

The calibration board was constructed by printing an A4 sized 11 $\times$ 8 chessboard pattern with a laser printer. The size of a single square can be determined digitally or measured after printing. Then a few papers were places below the printed pattern in order to smoothen the surface and to prevent patterns below the paper from showing. This stack of papers was mounted on a cardboard piece and held together with tape.

3.2 Calibration

The calibration begins by initializing the OpenCV video capture objects in Python. Autofocus is disabled from both cameras and the resolution is set to 640$\times$480. The cameras take pictures simultaneously and from each picture it is checked that a calibration pattern can be detected. Here, a checkerboard pattern is used as it is simple and easy to construct and offers flexibility. An 11$\times$8 pattern is chosen to provide enough object points for calibration. OpenCV may not be able to extract a too large pattern relative to the resolution from the image. Too small pattern, however, may not offer enough robustness to noise. Additionally, the pattern should be non-symmetric with even rows and odd columns in order to be unique defined in every direction. If a pattern is found from both cameras, the image is saved. Otherwise, new images are taken until a pattern is found. The calibration pattern is rotated and moved within the volume where the object localization is wanted. After successful 10–20 pictures are taken for reliability, the camera parameter calculation can be started.

The camera parameters are calculated according to the process described in [35]. Once the internal parameters are found, the re-projection error is calculated. If the camera calibration is successful, the intrinsic parameters K_1, K_2, d_1, d_2 can be saved, and the process described so far is no longer needed in the calibration phase. The extrinsic parameters of each camera $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{t_1}$ and $\mathbf{t_2}$ can be saved as well and used as long as neither camera moves or rotates. The implementation utilizes several OpenCV functions. First, the chessboard corners are found with a pixel level accuracy using `cv.findChessboardCorners`, and refined to sub-pixel accuracy with `cv.cornerSubPix`. Finally, the camera parameters are found with `cv.calibrateCamera`.

The next phase is to find the stereo parameters R and $\mathbf{t}$ relating the two cameras together from their individual extrinsic parameters R_1, R_2, $\mathbf{t_1}$ and $\mathbf{t_2}$. OpenCV returns the rotation as a vector, but it can be converted to the matrix form using equation 2.19 which is implemented in `cv.Rodriguez`. The relative extrinsic parameters can be then found with equations [26][32]

$$R = R_2 R_1 \tag{3.1}$$

and

$$\mathbf{t} = \mathbf{t_2} - R\mathbf{t_1}. \tag{3.2}$$

This approach is implemented in `cv.stereoCalibrate`. If the cameras have moved, the method in [35] can be used to find the extrinsic parameters while keeping the intrinsic parameters constant. These can be encoded into E, and F can be solved in this phase according to Equations 2.12 and 2.10. Once all parameters are found, they are combined into P_1 and P_2 according to equations

$$P_1 = K[I|\mathbf{0}] \tag{3.3}$$

and

$$P_2 = K[R|\mathbf{t}] \tag{3.4}$$

where P_1 is the first projection matrix at origin and P_2 is the second projection matrix.

This method does not require any more data if the cameras do not move between the intrinsic and extrinsic calibration. This method, that is called the calibration pattern method, for calculating the extrinsic parameters requires several pictures of the calibration pattern which is not ideal if the intrinsic parameters are already known.

This calibration pattern method is very sensitive to the accuracy of the intrinsic parameters and the accuracy of the detected corners. Therefore, additional method, that is called the keypoint method, is proposed. In this additional method, the extrinsic parameters are determined from just one image from both cameras. This method does not require a calibration pattern, and is based on feature extraction and the five-point algorithm. Because it does not require a calibration pattern, it is more flexible. However, the overall scale of the calibrated system is dependent on the intrinsic parameters. According to the original paper on five-point algorithm [23], it is enough to know the intrinsic parameters within 10 % of the actual values. Additionally, it is stated in the paper that with a higher error in calibration, uncalibrated methods such as the eight-point algorithm [5] out-perform the five-point algorithm.

Here, the first step is to extract keypoints from both images. The most straight forward method is to use the same calibration pattern and use the same method for recognizing the chessboard corners as used in camera calibration. As the same board is in both images, feature matching or descriptor extraction is not necessary. The corresponding keypoints are automatically in the same order because both images contain the same non-symmetric chessboard. Unlike the previous method, this method generalizes to a scene with no calibration pattern. In the general method keypoints and descriptors are extracted from both images. Here, the ORB algorithm is used. The ORB extracts a set of keypoints and descriptors from both images. Features are extracted from both images using `cv.ORB` class and its member function `detect`, the descriptors calculated using the member function `compute`, and are then matched using the nearest neighbor approach implemented in the class `cv.BFMatcher` and its `match` member function. The matching is based on the descriptors, and all low-enough distances are used. The threshold is

set empirically. From the matching phase the matching keypoints from both images are stored.

With these matching keypoints and the calibration matrix from the intrinsic calibration, the five-point algorithm can be invoked. The output of this is then the essential matrix E. This is implemented in OpenCV's `cv.findEssentialMat` function. It utilizes the RANSAC algorithm [7] in order to find the inlier matches. The essential matrix is only up to scale and has still four possible decompositions with the Equation 3.6. However, using the previously found matches, the correct signs can be determined. This is due to the cheirality constraint, which means that points seen by both cameras should be in front of both cameras. This can be checked by triangulating a point to 3D using all four combinations and checking whether the point is in front of both cameras. [23]

The next step in the keypoint method is to find the other extrinsic parameters. From E, F can be calculated with equation 2.10. From the essential matrix the extrinsic parameters R and $\mathbf{t}$ can be calculated using SVD (Singular Value Decomposition). Here, E can be factorized into

$$E = U\Sigma V^T \tag{3.5}$$

where Σ is a diagonal matrix of singular values of E, and U and V are orthogonal output matrices. From these, two possible combinations of translation and rotation can be recovered as

$$[\mathbf{t}]_x = UZU^T \quad \text{and} \quad R = UWV^T \quad \text{or} \quad R = UW^TV^T \tag{3.6}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.7}$$

Two additional combinations of R and $\mathbf{t}$ arise because the sign of E, and thus the sign of $\mathbf{t}$, is unknown. Hence, four projection matrices of the form of Equation 2.5 satisfy the epipolar constraint of Equation 2.11. However, only one of these is the correct projection matrix. [10][23]

Despite the ambiguity, this is a useful representation of the extrinsic parameters as this only has five degrees of freedom. With only five degrees of freedom the camera extrinsics are more easily calibrated. The correct one of the four possibilities can be identified with the cheirality test. Cheirality test means that triangulation yields to all points being in front of both cameras, or in other words, that the z-component of all in-line calibration points is positive. In OpenCV this is done with the function `cv.recoverPose`. The drawback of this method is that $\mathbf{t}$ is only known up to an arbitrary scale. The scale needs to be inferred from the scene, or by using some external information such as the physical separation of the cameras. For the chessboard case the method is straight forward. The distance between each camera and the chessboard is calculated, and using the knowledge of the previously

acquired relative rotation the angle between the cameras is known. Using the function `cv.solvePnP` the camera pose relative to known 3D points can be calculated. Then the translation with scale can be calculated with Equation 3.2. This requires knowing the positions of the 3D points relative to each other. In the chessboard case, the 3D points are one square length apart from each other. Then the length of $\mathbf{t}$ can be calculated by calculating its norm

$$t = \sqrt{t_1^2 + t_2^2 + t_3^2} \tag{3.8}$$

where t is the length of the relative translation and t_i are the components of the relative translation vector. When the physical separation is known, $\mathbf{t}$ can be scaled to the correct length by multiplying it with the physical separation.

From these the projection matrices for both cameras can be constructed. For simplicity, the origin of the common coordinate system is set at the center of one camera. Knowing the relative rotation, the relative translation, and the calibration matrix, the projection matrices can be constructed with Equations 3.3 and 3.4. With the projection matrices found, the camera set up is fully calibrated.

In order to evaluate the success of the calibration, the re-projection error is calculated. The re-projection error describes the root mean square (RMS) distance between original image points and first triangulated and then re-projected image points. The original set of image points is the detected calibration points, for example the chessboard corners. In this case the number of image points is limited to 50 in order to normalize the errors between different calibration methods. These are then triangulated into the 3D space with Algorithm 2.1 which is implemented in OpenCV's `cv.triangulatePoints`. Then the points are re-projected using `cv.projectPoints`. The error E_{rms} is calculated with the equation

$$E_{rms} = \sqrt{\frac{\sum_{i=0}^{N-1} \left(\mathbf{x_i} - \mathbf{x_i'}\right)^2}{N}} \tag{3.9}$$

where $\mathbf{x_i}$ is the original point, $\mathbf{x_i'}$ is the re-projected point, and N is the number of points.

3.3 Object Detection and Localization

Object detection is done with the COCO pre-trained Mask R-CNN. The network architecture and weights are downloaded from GitHub [1]. The network uses TensorFlow [6] back-end with Keras [4] interface. To be able to use this network directly in OpenCV with TensorFlow back-end, the model needs to be frozen. The script from the official TensorFlow repository [33] is used for freezing the network. In addition to the frozen model, the COCO labels are needed and they can be found from the original paper [18].

The frozen model is first loaded into OpenCV with `cv.dnn.readNetFromTensorflow`. When the OpenCV instance of the net is running, the model can be used to make predictions. This is done with the functions `cv.dnn.blobFromImage`, which pre-processes the image, then set as an input with the net instance's `setInput` function, and finally the predictions

are received with the net instance's `forward` member function. It should be noted that the original images that include distortion should be used as the input because undistortion might cause artifacts or discontinuities in the image [10]. The model returns both the masks and bounding boxes of the objects it detects. The masks can be omitted for this study, but the bounding boxes are collected and their positions in the image are calculated. Each bounding box is represented with four pixel values x_1, y_1, x_2 and y_2, which represent the corners of the bounding box. After the bounding box is detected, the effect of distortion should be removed. This is done with `cv.undistortPoints`, which implements the Equation 2.6. It should be noted that because the projection matrices were determined with respect to the image coordinates, undistortion should not normalize the points like OpenCV does by default. Therefore an identity matrix should be passed instead of the calibration matrix.

In an ideal case where the two cameras are close to each other and their principal axis are almost aligned, the matching points from the two images are determined via stereo rectification and stereo matching, and dense reconstruction is used. However, here feature matching is not possible because not all points are necessarily seen by both cameras, making the traditional stereo matching difficult and computing intensive. Therefore each corner of the bounding box from both images is back-projected into 3D space, and a bounding volume is calculated from intersections of the rays from the corners of the bounding boxes. For one corner, a pixel is triangulated into the 3D world using Algorithm 2.1 implemented in OpenCV's `cv.triangulatePoints`. The same is done for all pairs of bounding box corners until eight points in the 3D space are found. The pairs are formed by combining the top left-hand corner of the left-hand image with the top left-hand corner of the right image, the top left-hand corner of the left image with the top right-hand corner of the right image, and the same is done in reverse for the top corners. This is also repeated for the bottom corners.

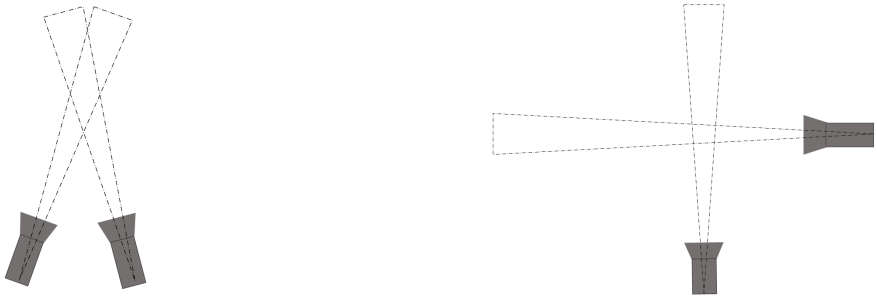


Figure 3.1. *Shapes of the bounding boxes when viewed from above.*

These triangulated points define the bounding box in 3D for the object. The bounding box will be in the shape of a truncated rectangle base pyramid, as shown in Figure 3.1. The tip of the pyramid is in the camera center of the left camera. The pyramid is truncated by a similar rectangle base pyramid whose tip is in the right camera center. This introduces a geometric constraint on the tightness of the bounding box. The closer the cameras are together, the larger the volume of the bounding box will be because the tips of the

pyramids are closer and hence only a small portion of the pyramid is truncated.

4 RESULTS

The success of the intrinsic parameter calibration can be calculated using the RMS error of re-projection. The RMS error was found to be 0.188 pixels. This result was obtained by simultaneously taking ten images of the same chessboard with both cameras, and taking the average of the two achieved re-projection errors. The chessboard was held close to the camera, and its location and angle were varied. Figure 4.1 shows an examples of such images. However, the result was obtained in a set up where the cameras were next to each other pointing in the same direction because then the chessboard covered more of the image. The optimum way would be to compare the calibrated focal length and principal point to the ground truth. However, this information was not available for this study because the camera manufacturer has not published that information.

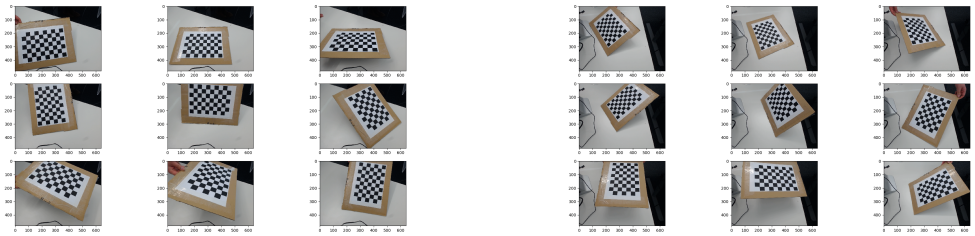


Figure 4.1. *Images for calibration.*

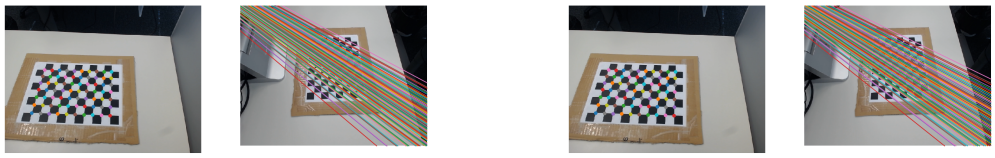


Figure 4.2. *A sample of the epipolar geometry visualized. The calibration pattern method on the left and the keypoint method on the right.*

Three different cases were considered and the position of the cameras was varied in order to study its effects on the resulting bounding box accuracy. The first image set shown in this chapter is when the cameras are at a right angle. The rest are in Appendix A where first the cameras are at an acute angle, and then next to each other. Because the area of interest was kept the same size, the distance between the cameras increases naturally as the angle of the principal axis increases.

The validity of extrinsic parameters is visualized by drawing the detected chessboard corners on one image and drawing the corresponding epipolar lines on the other. These are visualized first for the calibration pattern method, and secondly for the keypoint method. The results for one set up are shown in Figure 4.2. The results for the rest of the set up are presented in Figure A.1. As can be seen from the images, the accuracy of the two methods is comparable.

Table 4.1. *RMS re-projection errors.*

Image Set	RMS (calibration pattern)	RMS (keypoint)
1	0.585	0.594
2	0.644	0.612
3	0.621	0.626

Table 4.2. *Scales.*

Image Set	t (calibration pattern) / cm	t (keypoint) / cm	t (ground truth) / cm
1	28.35	29.97	30
2	22.41	23.76	22
3	11.84	14.29	12

The chessboard images can be projected back to the original calibration images to compare the re-projection accuracy. These are both again visualized for both methods in Figures 4.3 and A.2. These figures can also be summarized into one number, namely the back-projection error. This is done with Equation 3.9. The results are presented in Table 4.1. The effect of the translation is also significant to the absolute coordinates of the object, but not the relative coordinates. The absolute translations are listed in table 4.2.

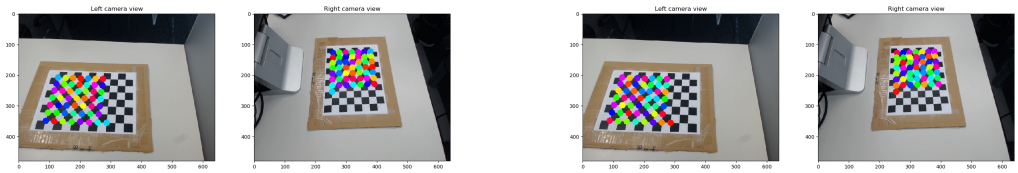


Figure 4.3. *A sample of the re-projected chessboards. The calibration pattern method on the left and the keypoint method on the right.*

Calculations of the bounding volume with both methods are illustrated in Figures 4.4 and A.3–A.4. The points in 3D are projected back to the original images as illustrated in Figures 4.5 and A.5. The projected points have similar colors on 2D and 3D. These figures also show the original bounding boxes.

The effect of noise was also studied. Gaussian noise was added to all calibration images as well as the test image containing the known object. It was noticed that both methods are quite robust to noise in the image. As long as the chessboard pattern can be detected,

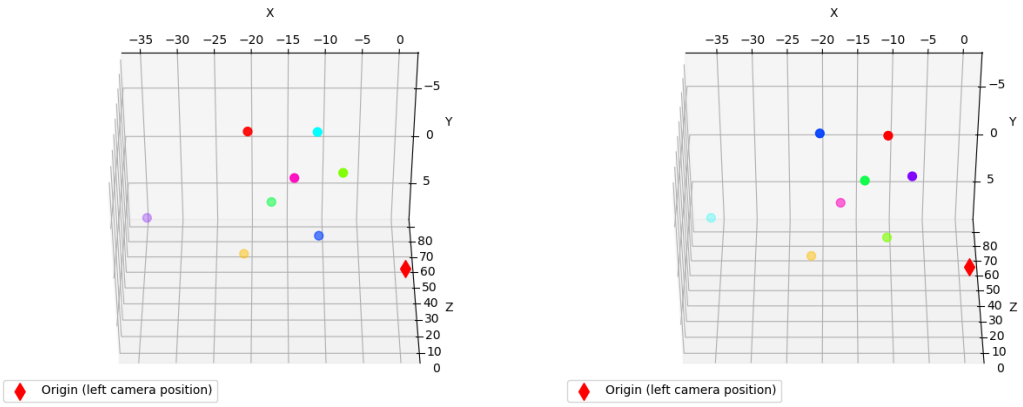


Figure 4.4. *The calculated bounding volume for right-angled views.*

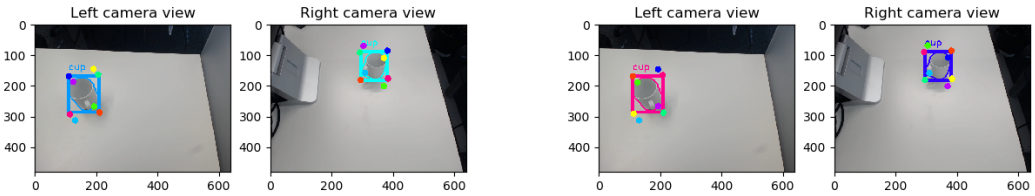


Figure 4.5. *The bounding box on both cameras.*

the re-projection accuracy remains practically the same. On the other hand, when similar Gaussian noise was added to the detected chessboard corner coordinates, both methods quickly became unusable, and there were no significant differences between the two methods. However, when outliers were introduced, differences started to show. Outliers were introduced by swapping to random image points in the calibration phase. It was noticed that the calibration pattern method does not tolerate any outliers. The keypoint method, on the other hand, tolerates several outliers, mainly due to running RANSAC before calibration. A realistic scenario where outliers may occur is when the calibration pattern is too small with respect to the image resolution. This in fact occurred at one point during testing with a board size of 20×20.

Figure 4.6. *Match-based calibration.*

Finally, an example of an extrinsic calibration without a calibration pattern using the ORB keypoint detector is presented. The calibration result and the found matches are presented in Figure 4.6, the 3D bounding box in Figure 4.7, and the 2D bounding box and a comparison to the same box but with the calibration pattern approach in Figure 4.8.

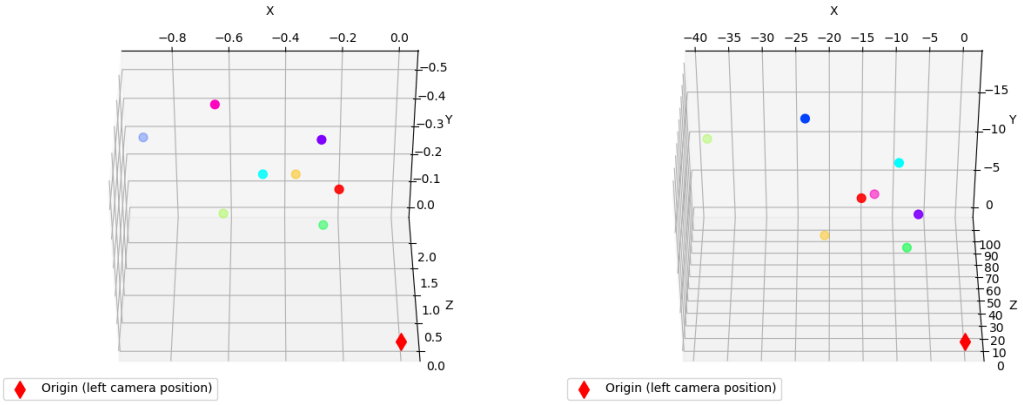


Figure 4.7. *The bounding boxes in 3D using the keypoint method. The matching method on the left, the pattern method on the right.*

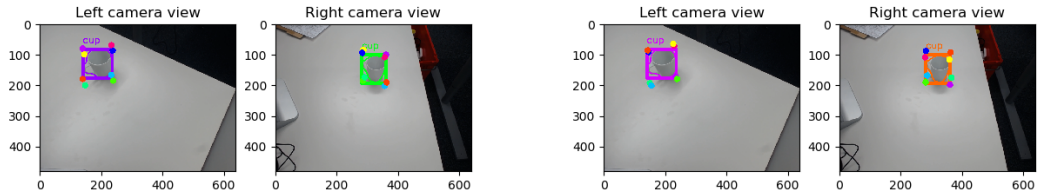


Figure 4.8. *The bounding box with automatic matches. The matching method on the left, the chessboard method on the right.*

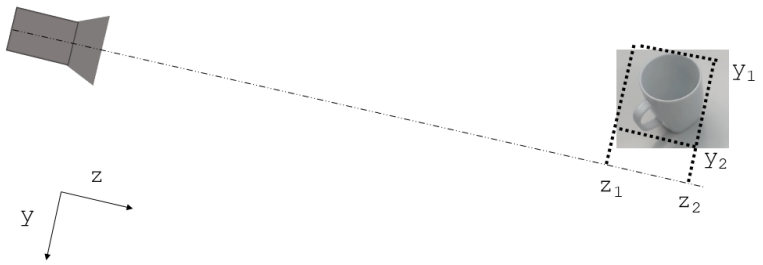


Figure 4.9. *A side view of the measurement set up.*

These bounding boxes were verified by measuring the actual location with a measuring tape. A side view of this of this method is shown in Figure 4.9. The x, y and z limits were estimated by measuring the perpendicular distances from the left camera where the origin lies. This method cannot provide ground truth values because it is not particularly

accurate. However, the method can be used to confirm that the results are approximately in the correct range. From the measurements it was noticed that the bounding boxes are in the same range that could be determined with a measuring tape. In the case where the two cameras are at a right angle, the 3D bounding box is the smallest due to geometric properties of the set up. Additionally, it can be seen that with several images of the chessboard, the estimated camera translation is more accurate than when estimated with a single image. This is because both methods use fundamentally the same approach but with multiple images the probability of random errors decreases, and also with multiple images there will be regression to the mean, which is mathematically the correct translation.

5 CONCLUSION

In this study a pipeline for sparse 3D reconstruction using two cameras is studied. The first step is to calibrate the camera intrinsic parameters. This is done using a planar calibration pattern with a known geometry. Then, the camera extrinsic parameters are calibrated which is done in two ways. The first method uses the same or similar calibration pattern as in the intrinsic calibration. The second method, on the other hand, infers keypoints, matches them, and finally estimates the external parameters with the five-point algorithm. This, however, requires some more knowledge of the scene, such as at least three known points for a unique solution. Once the cameras are calibrated, the 3D reconstruction phase can be started. Here the objects belonging to a known class are detected using Mask R-CNN but any other object detector that outputs bounding boxes would suffice. Then, the corners of the bounding boxes in both images are triangulated into the 3D world. The end result is a bounding box in 3D that is in the form of two intersecting rectangle based pyramids.

From the empirical results it can be seen that the calibration pattern approach and keypoint approach were comparable, and the calibration and bounding box accuracies are not affected Gaussian noise on the image. With similar Gaussian noise on the detected corner coordinates, the calibration accuracy decreases significantly even when small noise is present. On the other hand, when outliers were introduced, the performance of the calibration pattern method understandably decreases. Conversely, the accuracy of the keypoint method stays the same, however, it uses RANSAC to reduce outliers. Realistically outliers may be introduced when the calibration pattern is not properly detected. Additionally, the keypoint approach makes it possible to have a larger scene. Constructing a large enough calibration pattern for cameras separated by tens of meters would be difficult and expensive. However, by detecting matches from the scene, no calibration pattern is needed.

The overall results were satisfying for the use case. The location of the bounding box was accurate within centimeters in this case. When the system is scaled into large scenes, the error naturally increases. However, relative locations should be possible to be determined with this set up. When there are several objects, additional constraints could be added to aid the localization. The results suggest that when the cameras are looking at the scene from a relatively high position, the optimal set up would be to set the cameras at a large separation and steep angle with respect to each other. This could be further improved by adding a third camera, and setting the cameras at equal angles to each other. This would

also help with the problems introduced by occlusions. Another way the bounding boxes could be restricted is by detecting the ground level and assuming all objects are on the ground plane. Then only the portion of the bounding box that is above the ground plane could be accounted.

REFERENCES

[1] W. Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. Online. Accessed: 2019-02-23. 2017. URL: https://github.com/matterport/Mask_RCNN.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

[3] M. Calonder, V. Lepetit & P. Strecha Christophand Fua. BRIEF: Binary Robust Independent Elementary Features. *European Conference on Computer Vision (ECCV)*. Ed. by K. Daniilidis, P. Maragos & N. Paragios. Berlin, Heidelberg: Springer, 2010, 778–792.

[4] F. Chollet et al. *Keras*. Online. Accessed: 2019-02-23. 2015. URL: https://keras.io.

[5] H. Christopher Longuet-Higgins. A Computer Alorithm for Reconstructing a Scene from Two Projections. *Nature*. Vol. 293. Sept. 1981, 133–135.

[6] J. Dean, R. Monga et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[7] M. A. Fischler & R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*. Vol. 24. 6. New York, NY, USA: ACM, June 1981, 381–395.

[8] R. Girshick. Fast R-CNN. *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[9] C. Harris & M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*. 1988, 147–151.

[10] R. Hartley & A. Zisserman. *Multiple view geometry in computer vision*. 2nd. New York, NY, USA: Cambridge University Press, 2003.

[11] K. He, G. Gkioxari, P. Dollar & R. Girshick. Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017.

[12] E. Jones, T. Oliphant, P. Peterson et al. *SciPy: Open source scientific tools for Python*. Online. Accessed: 2019-02-22. 2001. URL: http://www.scipy.org/.

[13] A. Kaehler & G. R. Bradski. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. 1st. OReilly, 2016.

[14] A. Krizhevsky, I. Sutskever & G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. Vol. 25. Jan. 2012.

[15] Y.-J. Lee & M.-W. Park. 3D tracking of multiple onsite workers based on stereo vision. *Automation in Construction*. Vol. 98. 2019, 146–159.

[16] H. Li & R. I. Hartley. Five-Point Motion Estimation Made Easy. Vol. 1. 2006, 630–633.

[17] P. Li, T. Qin & S. Shen. Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving. *European Conference on Computer Vision (ECCV)*. Ed. by V. Ferrari, M. Hebert, C. Sminchisescu & Y. Weiss. Cham: Springer International Publishing, 2018, 664–679.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár & C. L. Zitnick. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science*. Springer International Publishing, 2014, 740–755.

[19] S. Liu, Y. Liu, Z. Wang & Y. Cao. Automatic chessboard corner detection method. *IET Image Processing*. Vol. 10. Sept. 2015.

[20] J. Long, E. Shelhamer & T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[21] B. Mellish. *How a pinhole camera works.* Online. Licenced under CC BY-SA (`https://creativecommons.org/licenses/by-sa/3.0/deed.en`) Accessed: 2019-05-04. 2005. URL: `https://commons.wikimedia.org/wiki/File:Pinhole-camera.png`.

[22] J. Morat, F. Devernay, J. Ibanez-Guzman & S. Cornou. Evaluation Method for Automotive Stereo-Vision Systems. *IEEE Intelligent Vehicles Symposium*. June 2007, 202–208.

[23] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 26. 6. June 2004, 756–770.

[24] D. Nister. Preemptive RANSAC for live structure and motion estimation. *IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. Oct. 2003, 199–206.

[25] A. Nordmann. *Epipolar geometry*. Online. Lincenced under CC BY-SA (`https://creativecommons.org/licenses/by-sa/3.0/deed.en`). Accessed: 2019-03-26. 2007. URL: `https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg`.

[26] *OpenCV calib3d module documentation. Camera Calibration and 3D Reconstruction*. Version 4.0.0. Online. Accessed: 2019-02-19. URL: `https://docs.opencv.org/4.0.0/d9/d0c/group__calib3d.html`.

[27] S. Ren, K. He, R. Girshick & J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama & R. Garnett. Curran Associates, Inc., 2015, 91–99.

[28] P. L. Rosin. Measuring Corner Properties. *Computer Vision and Image Understanding*. Vol. 73. 1997, 291–307.

[29] G. Rossum. Python Reference Manual. Amsterdam, Alankomaat: Centre for Mathematics and Computer Science, 1995.

[30] E. Rosten & T. Drummond. Machine Learning for High-Speed Corner Detection. *European Conference on Computer Vision (ECCV)*. Ed. by A. Leonardis, H. Bischof & A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 430–443.

[31] E. Rublee, V. Rabaud, K. Konolige & G. Bradski. ORB: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision (ICCV)*. Nov. 2011, 2564–2571.

[32] R. Szeliski. *Computer Vision: Algorithms and Applications*. 2010 draft. 2010. URL: http://szeliski.org/Book/.

[33] *TensorFlow. An Open Source Machine Learning Framework for Everyone*. Online. Accessed: 2019-02-23. 2015. URL: https://github.com/tensorflow/tensorflow.

[34] V. Vladimir. *OpenCV calibration object detection*. Online. Accessed: 2019-03-24. June 21, 2004. URL: https://graphicon.ru/oldgr/en/research/calibration/opencv.html.

[35] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22. 11. 2000, 1330–1334.

A FURTHER RESULTS

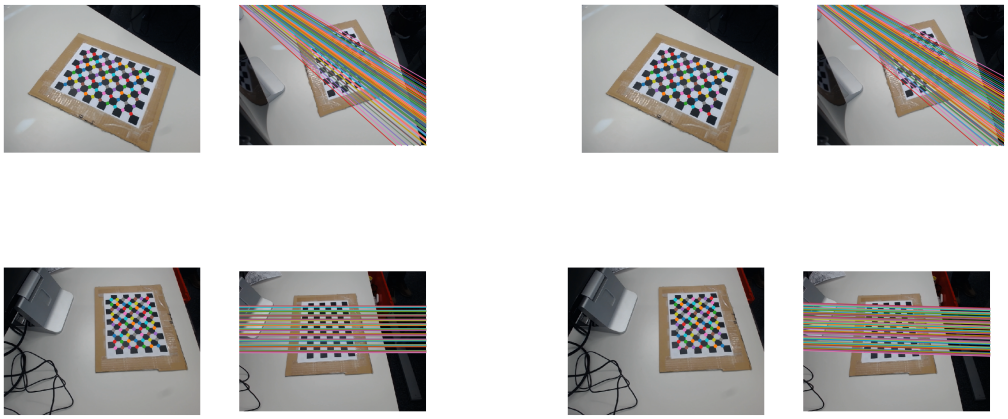


Figure A.1. *Rest of the epipolar geometry visualized. The calibration pattern method on the left and the keypoint method on the right.*

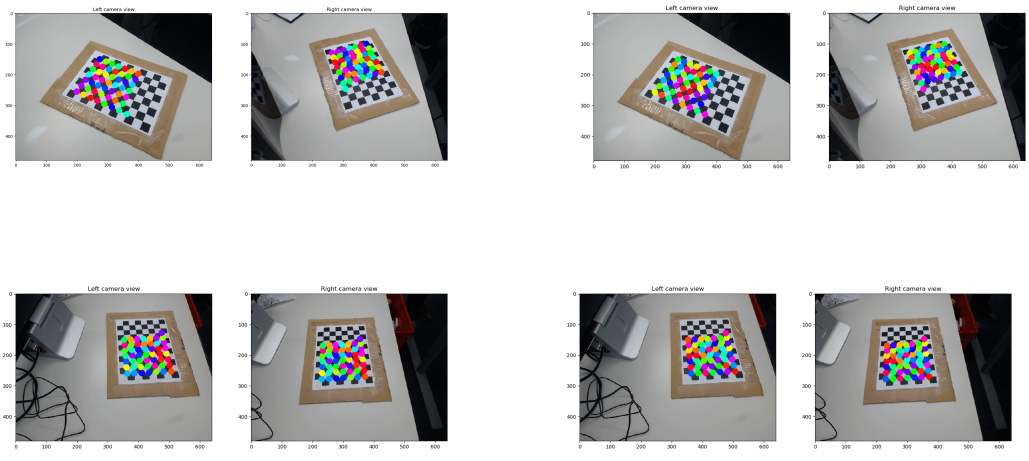


Figure A.2. *Rest of the re-projected chessboards. The calibration pattern method on the left and the keypoint method on the right.*

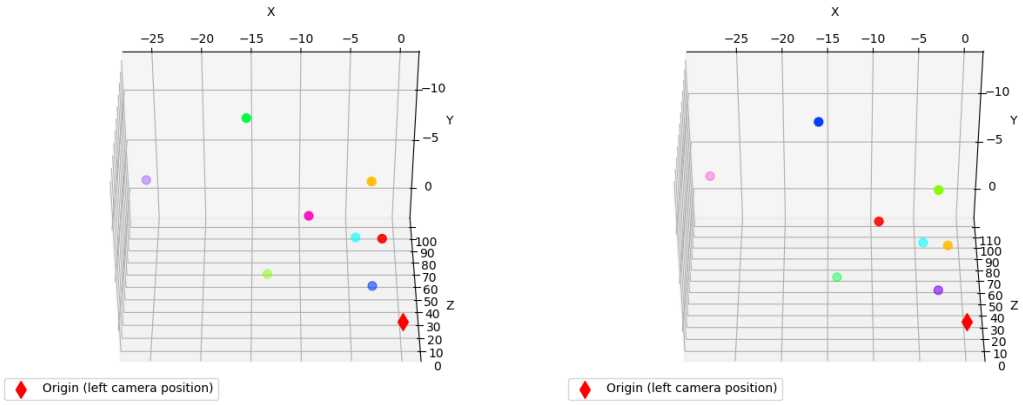


Figure A.3. *The calculated bounding volume for acute angled views. The calibration pattern method on the left and the keypoint method on the right.*

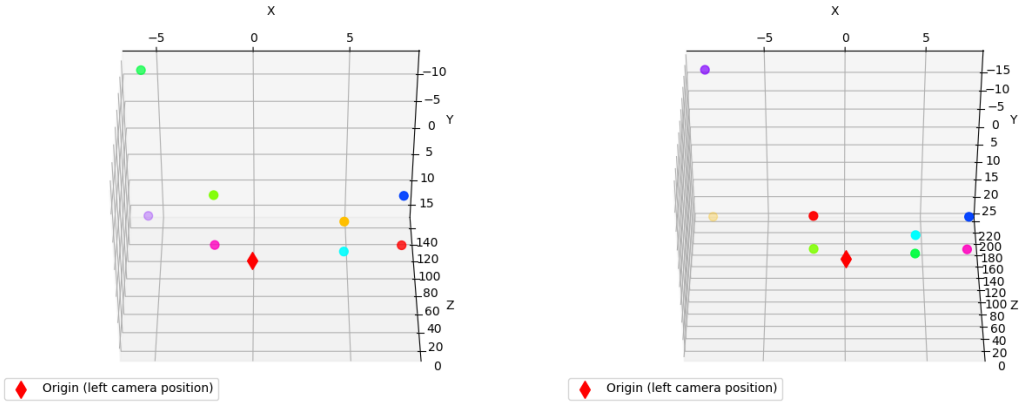


Figure A.4. *The calculated bounding volume for similar views. The calibration pattern method on the left and the keypoint method on the right.*

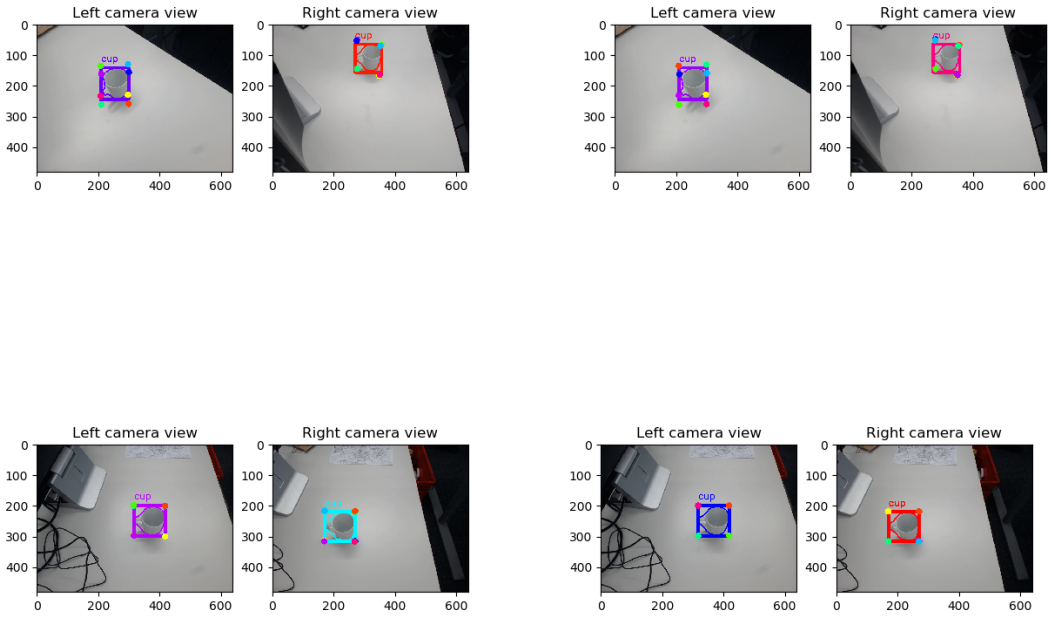


Figure A.5. *The rest of the bounding boxes on both camera views.*